

In the Claims:

Please amend claims 48 and 60 as indicated below:

1. (Previously presented) A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement an error trace mechanism for a multithreaded program configured to:

in each thread of the multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in a memory storage area specific to the thread in accordance with an application programming interface (API) to the error trace mechanism; and

obtain an error trace for each thread of the multithreaded program in accordance with the API to the error trace mechanism;

wherein each error trace includes one or more error trace elements specific to the corresponding thread, wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

2. – 3. (Canceled)

4. (Previously presented) The system as recited in claim 1, wherein the error trace further includes a field indicating a count of the error trace elements in the error trace.

5. (Original) The system as recited in claim 1, wherein each error trace element indicates one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

6. (Original) The system as recited in claim 5, wherein the location of the particular error includes one or more of a function name, a source file name, and a line number where the particular error occurred.

7. (Previously presented) The system as recited in claim 1, wherein the program is further configured to determine from each error trace element one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

8. (Previously presented) The system as recited in claim 1, wherein the error trace mechanism is a C/C++ interface library.

9. (Previously presented) A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement a library and a multithreaded program configured to call library functions of the library in accordance with an application programming interface (API) to the library;

wherein the library is configured to, for each thread of the multithreaded program, add an error trace element for each error generated on the thread by the library functions to an error trace in a memory storage area specific to the thread, wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread;

and

wherein, after completion of a called library function, the multithreaded program is configured to obtain an error trace for a thread corresponding to the call of the library function in accordance with the API to the library.

10. (Previously presented) The system as recited in claim 9, wherein the called library function is configured to call one or more other library functions in a function call stack, wherein each of the one or more other library functions is configured to, if the library function generates an error, add an error trace element to an error trace in a memory storage area specific to a thread corresponding to the function call stack.

11. (Previously presented) The system as recited in claim 9, wherein the multithreaded program is further configured to determine from the error trace element one or more of a location where each error occurred, an error type of each error, and what each error is.

12. (Original) The system as recited in claim 9, wherein the library is a C/C++ interface library.

13. – 30. (Canceled)

31. (Previously presented) A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement a library comprising one or more library functions and an application programming interface (API) to the library, wherein the API includes:

one or more function definitions configured for access of the one or more library functions by a multithreaded program; and

a function definition for a get error trace function configured for access by the multithreaded program to get error traces generated by the one or more library functions in two or more threads of the multithreaded program, wherein each error trace is stored in a memory storage area specific to the thread;

wherein each error trace includes one or more error trace elements specific to the corresponding thread, wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

32. (Previously presented) The system as recited in claim 31, wherein one of the library functions is configured to call a plurality of other library functions in a function call stack, wherein each of the plurality of library functions is configured to, if the library function generates an error, add an error trace element to the error trace in a memory storage area specific to a thread corresponding to the function call stack.

33. (Original) The system as recited in claim 32, wherein the location of the particular error includes one or more of a function name, a source file name, and a line number where the particular error occurred.

34. (Previously presented) The system as recited in claim 31, wherein the multithreaded program is configured to determine from each error trace element one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

35. (Original) The system as recited in claim 31, wherein the library is a C/C++ interface library.

36. (Previously presented) A system, comprising:

means for a plurality of functions in each of two or more function call stacks in a multithreaded program to generate information describing one or more errors generated by the plurality of functions;

means to store the generated information in memory storage areas specific to threads corresponding to the function call stacks;

means to obtain the generated information; and

means to determine from the obtained information one or more of a location where each error occurred, an error type of each error, and what the each error is.

37. (Previously presented) The system as recited in claim 36, wherein the plurality of functions are functions of a library, further comprising means to call the plurality of functions from the multithreaded program.

38. (Original) The system as recited in claim 37, wherein the library is a C/C++ interface library.

39. (Previously presented) A method, comprising:

calling one or more functions in each of two or more threads of a multithreaded program;

in each thread of the multithreaded program, for each error generated by the one or more functions called in the thread, storing an error trace element in a memory storage area specific to the thread in accordance with an

application programming interface (API) to an error trace mechanism; and

the program obtaining an error trace for each thread of the multithreaded program in accordance with the API to the error trace mechanism;

wherein each error trace includes one or more error trace elements specific to the corresponding thread, wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

40. – 41. (Canceled)

42. (Original) The method as recited in claim 39, wherein each error trace element indicates one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

43. (Original) The method as recited in claim 42, wherein the location of the particular error includes one or more of a function name, a source file name, and a line number where the particular error occurred.

44. (Currently amended) The method as recited in claim 39, further comprising determining from each error trace element one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

45. (Canceled)

46. (Previously presented) The method as recited in claim 39, wherein the error trace mechanism is a C/C++ interface library.

47. (Previously presented) A method, comprising:

a multithreaded program calling library functions of a library in accordance with an application programming interface (API) to the library;

for each thread of the multithreaded program, adding an error trace element for each error generated on the thread by the library functions to an error trace in a memory storage area specific to the thread;

after completion of a called library function, the multithreaded program obtaining an error trace for a thread corresponding to the call of the library function in accordance with the API to the library;

wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

48. (Currently amended) The method as recited in claim 47, further comprising:

the called library function calling one or more other library functions in a function call stack; and

for each of [[the]] the one or more other library functions, if the library function generates an error, adding an error trace element to the error trace in a memory storage area specific to a thread corresponding to the function call stack.

49. (Original) The method as recited in claim 47, further comprising determining from the error trace element one or more of a location where each error occurred, an error type of each error, and what each error is.

50. (Original) The method as recited in claim 47, wherein the library is a C/C++ interface library.

51. (Previously presented) A computer-accessible storage medium, comprising program instructions, wherein the program instructions are computer-executable to implement:

calling one or more functions in each of two or more threads of a multithreaded program;

in each thread of the multithreaded program, for each error generated by the one or more functions called in the thread, storing an error trace element in a memory storage area specific to the thread in accordance with an application programming interface (API) to an error trace mechanism; and

the program obtaining an error trace for each thread of the multithreaded program in accordance with the API to the error trace mechanism;

wherein each error trace includes one or more error trace elements specific to the corresponding thread, wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

52. – 53. (Canceled)

54. (Previously presented) The computer-accessible storage medium as recited in claim 51, wherein each error trace element indicates one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

55. (Previously presented) The computer-accessible storage medium as recited in claim 54, wherein the location of the particular error includes one or more of a function name, a source file name, and a line number where the particular error occurred.

56. (Previously presented) The computer-accessible storage medium as recited in claim 51, wherein the program instructions are further computer-executable to implement determining from each error trace element one or more of a location where the particular error of the error trace element occurred, an error type of the particular error, and what the particular error is.

57. (Canceled)

58. (Previously presented) The computer-accessible storage medium as recited in claim 51, wherein the error trace mechanism is a C/C++ interface library.

59. (Previously presented) A computer-accessible storage medium, comprising program instructions, wherein the program instructions are computer-executable to implement:

a multithreaded program calling library functions of a library in accordance with an application programming interface (API) to the library;

for each thread of the multithreaded program, adding an error trace element for each error generated on the thread by the library functions to an error trace in a memory storage area specific to the thread;

after completion of a called library function, the multithreaded program obtaining an error trace for a thread corresponding to the call of the library function in accordance with the API to the library;

wherein each error trace element includes information describing a particular error generated during execution of the corresponding thread.

60. (Currently amended) The computer-accessible storage medium as recited in

claim 59, wherein the program instructions are further computer-executable to implement:

the called library function calling one or more other library functions in a function call stack; and

for each of [[the]] the one or more other library functions, if the library function generates an error, adding an error trace element to the error trace in a memory storage area specific to a thread corresponding to the function call stack.

61. (Previously presented) The computer-accessible storage medium as recited in claim 59, wherein the program instructions are further computer-executable to implement determining from the error trace element one or more of a location where each error occurred, an error type of each error, and what each error is.

62. (Previously presented) The computer-accessible storage medium as recited in claim 59, wherein the library is a C/C++ interface library.